BINUS UNIVERSITY ONLINE LEARNING

People
Innovation
Excellence

LECTURE NOTES

Algorithm and Programming

Minggu 7

Structures and Union

LEARNING OUTCOMES

LO 2: Menganalisis program untuk memecahkan masalah menggunakan bahasa C.

Outcome:

Mahasiswa mampu menganalisis program untuk memecahkan masalah menggunakan bahasa C.

OUTLINE MATERI (Sub-Topic):

- Structure
- Typedef
- Union
- Bit Field
- Enumeration

ISI MATERI

1. Structure

Structure atau struct merupakan struktur data yang menggabungkan beberapa data yang berbeda tipe (heterogen) tetapi data tersebut saling berkaitan. Misalnya data mengenai StudentId, nama, dan IPK seorang mahasiswa. Ketiga data ini memiliki tipe data yang berbeda tetapi masih saling berhubungan yaitu data akademik seorang mahasiswa. Dengan megngunakan struct maka data ini bisa diolah per-elemen (per field) atau secara keseluruhan (per struct, per record).

Untuk mendefinisikan struktur, Anda harus menggunakan pernyataan struct. Pernyataan Struct mendefinisikan tipe data baru, dengan lebih dari satu anggota. Format pernyataan struct adalah sebagai berikut -

Berikut contoh sintaks struct pada bahasa C :

struct [type_name]

{

member definition; member definition;

...

member definition;

} [one or more structure variables];

Type_name : adalah nama tipe struct.

Member definition: adalah nama member atau field.

one or more structure variables: adalah nama variabel struct. Berikut adalah beberapa contoh deklarasi struct pada bahasa C :

```c
//deklarasi variabel struct.
struct {
    char StudentId[11];
    float ipk;
    char nama[50];
}mhs;
```

```c
//deklarasi tipe struct dan variabel secara bersamaan.
struct data{
    char StudentId[11];
    float ipk;
    char nama[50];
}mhs;
```

```c
//deklarasi tipe struct dan variabel struct secara terpisah.
struct data{
    char StudentId[11];
    float ipk;
    char nama[50];
};

struct data mhs;
```

Pada beberapa compiler C, keyword struct pada deklarasi variabel struct secara terpisah dapat dihilangkan. Seperti tipe data yang lain, sebuah variabel struct dapat diinisialisasi diawal pada saat pendeklarasian. Berikut contoh inisialisasi struct pada C:

```c
#include<stdio.h>

struct data{
    char StudentId[11];
    float ipk;
    char nama[50];
};

int main()
{
    struct data mhs = {"1900012223", 3.78, "Adam Syahputra"};
    printf("StudentId : %s\n", mhs.StudentId);
    printf("IPK : %.2f\n", mhs.ipk);
    printf("Nama : %s\n", mhs.nama);

  getchar();
  return 0;
}
```

Untuk mengakses field struct gunakan dotoperator yaitu nama variabel struct diikuti titik dan nama field. Berikut adalah hasil output dari contoh inisialisasi sturct diatas :

```
StudentId : 1900012223
IPK : 3.78
Nama : Adam Syahputra
```

2. Typedef

Pada bahasa pemrograman C disediakan keyword typedef, yang dapat digunakan untuk membuat tipe penamaan baru. Misalnya typedef dapat digunakan dengan struct untuk mendefinisikan tipe data baru dan kemudian tipe data ini digunakan untuk mendefinisikan variabel struct, dengan contoh sebagai berikut :

```c
#include <stdio.h>
#include <string.h>
typedef struct Books
{
   char  title[50];
   int   book_id;
} Book;      //variable sturct

int main( )
{
   Book book;   //struct name
   strcpy( book.title, "C Programming");
   book.book_id = 6495407;

   printf( "Book title : %s\n", book.title);
   printf( "Book book_id : %d\n", book.book_id);

getchar();
return 0;
}
```

Berikut output dari program tersebut adalah :

```
Book title : C Programming
Book book_id : 6495407
```

Menggunakan Typedefs dapat membantu membuat program lebih mudah dibaca dan dipelihara. Seringkali, typedef digunakan untuk membuat sinonim untuk tipe bawaan. Misalnya, program yang membutuhkan bilangan bulat empat byte dapat menggunakan tipe *int* pada satu sistem dan tipe *long* pada yang lain. Program yang dirancang untuk portabilitas sering menggunakan typedef untuk membuat alias untuk bilangan bulat empat byte, seperti integer. Untuk port program ke platform lain, Anda dapat dengan mudah mengubah integer typedef dan mengkompilasi ulang program.

3. Union

Seperti struktur, union adalah tipe data yang diturunkan, tetapi anggotanya berbagi memori yang sama. Pada waktu yang berbeda selama eksekusi program, beberapa variabel mungkin tidak relevan. Jadi, Union berbagi ruang daripada membuang -buang penyimpanan pada variabel yang tidak digunakan. Anggota union dapat memiliki jenis apa pun. Jumlah byte yang digunakan untuk menyimpan union harus setidaknya cukup untuk menahan anggota terbesarnya.

Dalam kebanyakan kasus, union berisi dua atau lebih item dari berbagai jenis. Anda hanya dapat merujuk satu anggota - dan dengan demikian hanya satu jenis -

pada satu waktu. Adalah tanggung jawab Anda untuk merujuk data dengan jenis yang tepat. Merujuk data yang saat ini disimpan dengan variabel dari tipe yang salah adalah kesalahan logika-hasilnya tergantung pada implementasi.

Berikurt sintaks dalam penggunaan union pada bahasa C :

union [type_name]

{

member definition;

member definition;

...

member definition;

} [one or more union variables];

Type_name : adalah nama tipe union.

Member definition : adalah nama-nama variabel yang kongsi memori, minimal memiliki 2 variabel.

one or more union variables : adalah nama variabel union. Berikut contoh penggunaan union pada bahasa C :

```c
#include <stdio.h>
#include <string.h>

union Data
{
   int i;
   float f;
   char  str[20];
};

int main( )
{
   union Data data;

   printf( "Memory size occupied by data : %d\n", sizeof(data));

getchar();
return 0;
}
```

Pada contoh program tersebut terlihat bahwa memori terbesar adalah charstr sebesar 20 byte maka memori yang dibutuhkan adalah sebesar 20 byte. Dari program tersebut akan menghasilkan output sebagai berikut :

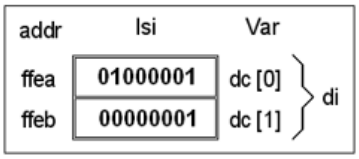
```
Memory size occupied by data : 20
```

Contoh

```
# include <stdio.h>

void main() {
  union tbil{
    unsigned int di;
    unsigned char dc[2];
  } bil_x;

  bil_x.di = 321;
  printf("di    = %d \n", bil_x.di);
  printf("dc[0] = %d \n", bil_x.dc[0]);
  printf("dc[1] = %d\n",  bil_x.dc[1]);
}
//size assumption for int data type is  2 bytes
```




```
di    = 321
dc[0] = 65
dc[1] = 1
```

Hasil :

321= 101000001

Contoh program untuk mengonversi 1 byte angka desimal (0-255) menjadi sistem biner menggunakan Union dan Bit-Field

```
#include <stdio.h>
struct biner{
      unsigned bit0:1; unsigned bit1:1;
      unsigned bit2:1; unsigned bit3:1;
      unsigned bit4:1; unsigned bit5:1;
      unsigned bit6:1; unsigned bit7:1;
};
int main()
{
   unsigned char ch;
   union byte x;
   printf("Input number (0-255): ");
   scanf("%d",&ch);
   x.ch=ch;
   printf("%d binary = %d%d%d%d%d%d%d%d\n",
   ch,x.bit.bit7,x.bit.bit6,x.bit.bit5,x.bit.bit4,
```

```
union byte{
      unsigned char ch;
      struct biner bit;
};
```

```
    x.bit.bit3,x.bit.bit2,x.bit.bit1,x.bit.bit0);
    getch();
    return 0;
}
```

Operasi yang dapat dilakukan pada union adalah:

- menugaskan union ke union lain dari jenis yang sama,
- Mengambil alamat variabel union (&),
- mengakses anggota union melalui operator anggota struktur (.) dan operator pointer struktur (->), dan
- Zero-initialized Union.

Dua union tidak boleh dibandingkan menggunakan operator == dan! = Untuk alasan yang sama bahwa dua struktur tidak dapat dibandingkan.

4. Bit Field

Anda dapat menentukan jumlah bit untuk menyimpan anggota integral yang unsigned atau signed dari sebuah struct atau union. Dikenal sebagai **bit field**, ini memungkinkan pemanfaatan memori yang lebih baik dengan menyimpan data dalam jumlah minimum bit yang diperlukan. Bit Field biasanya dinyatakan sebagai int atau unsigned int.

Contoh

```
struct bitCard {
    unsigned int face : 4;
    unsigned int suit : 2;
    unsigned int color : 1;
};
```

Array layar diisi dengan 2000 elemen, di mana setiap elemen menggunakan 2 byte (14 bit membutuhkan setidaknya dua byte)

Bit field dapat mengurangi jumlah memori yang dibutuhkan program, tetapi bergantung pada mesin. Meskipun Bit field menghemat ruang, menggunakannya dapat menyebabkan kompiler menghasilkan kode bahasa mesin yang lebih lambat. Ini terjadi karena dibutuhkan operasi bahasa mesin tambahan untuk mengakses hanya bagian dari unit penyimpanan yang dapat dialamatkan.

Bit field tidak memiliki alamat, jadi berusaha mengambil alamat bit field dengan "&" operator adalah kesalahan. Juga, menggunakan *sizeof* dengan bit field adalah kesalahan.

unnamed bit field digunakan sebagai padding dalam struct. Misalnya,

```
struct example {
    unsigned int a : 13;
    unsigned int : 19;
    unsigned int b : 4;
};
```

menggunakan field 19-bit yang unnamed sebagai padding. Tidak ada yang bisa disimpan dalam 19-bit itu. Anggota B (dengan asumsi komputer four-byte-word) disimpan dalam kata memori yang terpisah.

Bidang bit yang tidak disebutkan namanya dengan lebar nol menyelaraskan bidang bit berikutnya pada batas unit penyimpanan baru. Misalnya, struct.

```
struct example {
unsigned int a : 13;
unsigned int : 0;
unsigned int : 4;
};
```

menggunakan unnamed field 0-bit untuk melewatkan bit yang tersisa (sebanyak yang ada) dari unit penyimpanan di mana A disimpan dan untuk menyelaraskan B pada batas unit penyimpanan berikutnya

5. Enumeration

Enumeration adalah tipe data yang ditetapkan oleh pengguna yang terdiri dari konstanta integral dan setiap konstanta tersebut diberikan nama. Berikut sintaks enum pada bahasa C :

enumtype_name{ value1, value2,...,valueN };

Type_name : adalah nama dari tipe data enum.

Value : nilai dari enum

Berikut adalah contoh penggunaan enum pada bahasa C :

```c
#include <stdio.h>
#include <string.h>

enum week{ sunday, monday, tuesday, wednesday, thursday, friday, saturday};

int main()
{
    enum week today;
    today=wednesday;
    printf("%d day",today+1);

getchar();
return 0;
}
```

Berikut hasil dari output program diatas :

```
4 day_
```

Contoh

```c
#include <stdio.h>
enum boolean {false, true};

enum boolean evenCheck(int n) {
  enum boolean checkResult;
  if (n % 2  == 0) checkResult = true;
  else checkResult = false;
  return checkResult;
}

int main () {
  int num;
  enum boolean result;

  scanf("%d", &bil);
  result = evenCheck(bil);
  if (result == true) printf(“even");
  else printf(“odd");
  return 1;
}
```

```
100
even
```

```
37
odd
```

.

SIMPULAN

- Structure atau struct merupakan struktur data yang menggabungkan beberapa data yang berbeda tipe (heterogen) tetapi data tersebut saling berkaitan.
- Typedef, yang dapat digunakan untuk membuat tipe penamaan baru. Misalnya typedef dapat digunakan dengan struct untuk mendefinisikan tipe data baru dan kemudian tipe data ini digunakan untuk mendefinisikan variabel struct.
- Union digunakan untuk kongsi (menggabungkan) memori. Dengan menggunakan union, suatu lokasi memori dapat ditempati oleh dua atau beberapa variabel dengan masing-masing tipe data yang berbeda.
- Anda dapat menentukan jumlah bit untuk menyimpan anggota integral yang unsigned atau signed dari sebuah struct atau union. Dikenal sebagai **bit field**, ini memungkinkan pemanfaatan memori yang lebih baik dengan menyimpan data dalam jumlah minimum bit yang diperlukan.
- Enumeration adalah tipe data yang ditetapkan oleh pengguna yang terdiri dari konstanta integral dan setiap konstanta tersebut diberikan nama.

DAFTAR PUSTAKA

- Paul J. Deitel & Harvey. Deitel. (2022). C how to program. 09. Pearson Education. Hoboken. ISBN:978-0-13-739839-3. Chapter 10
- Collecting Data Items of Different Types: http://aelinik.free.fr/c/ch19.htm
- Structs, Enums, and Unions: http://www.lysator.liu.se/c/c-faq/c-9.html
- http://www.tutorialspoint.com/cprogramming/c_structures.htm
- http://www.tutorialspoint.com/cprogramming/c_typedef.htm
- http://www.tutorialspoint.com/cprogramming/c_unions.htm
- http://www.programiz.com/c-programming/c-enumeration